

1 **Analysis of College Completion at Institutions in the United States**

2
3 GOUTHAM CHADALLA MANJUNATH, UBIT-gouthamc UB-Number-50466616

4
5 MOHIT MAHESH VASWANI, UBIT-mohitmh UB-Number-50468309

6 SHREYAS ATHREYA VENKATESH, UBIT-venkate7 UB-Number-50366325

7
8
9 **ACM Reference Format:**

10 Goutham Chadalla Manjunath, Mohit Mahesh Vaswani, and Shreyas Athreya Venkatesh. 2022. Analysis of College Completion at
11 Institutions in the United States. 1, 1 (December 2022), 20 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

12
13
14 **1 PROBLEM STATEMENT**

15
16 The data-set that we have chosen for this project is College Completion dataset. The purpose of selecting this dataset is
17 that students would make informed decision before choosing a degree program based on the degree completion rates in
18 various universities. Additionally, Universities could use this dataset to improve their degree programs, understand
19 the demographic of the students. Universities could also make changes to their degree offering based on the student
20 completion rates of specific degrees. Furthermore, the federal and State education departments can take note of the
21 demographic data and make appropriate resource allocation to uplift those Universities with greater dropout rates.
22
23 A database system supports multiple tables whereas excel files have limited support for the number of tables. A database
24 supports the integrity of data whereas in excel files it is limited. Storing data in a database, therefore, promotes data
25 being stored centrally which in turn leads to data consistency

26
27
28 **2 TARGET USER**

- 29
30 • The User - The Main targeted user for this information are students that would be embarking on their journey
31 for a degree program. This data would help such students make an informed decision on what degrees to choose
32 from, and which university to settle on based on the graduation rates.
- 33
34 • The administers - The administer will be the universities/University board/ websites like US News. Additionally,
35 the Education department of the Federal and State government could use such a database for better resource
36 allocation.
- 37
38 • Real-life scenario - Situations where students wish to join the universities which provide a maximum probability
39 of success rate combined with location preference.

40
41 Authors' addresses: Goutham Chadalla Manjunath, UBIT-gouthamc UB-Number-50466616; Mohit Mahesh Vaswani, UBIT-mohitmh UB-Number-50468309;
42 Shreyas Athreya Venkatesh, UBIT-venkate7 UB-Number-50366325.

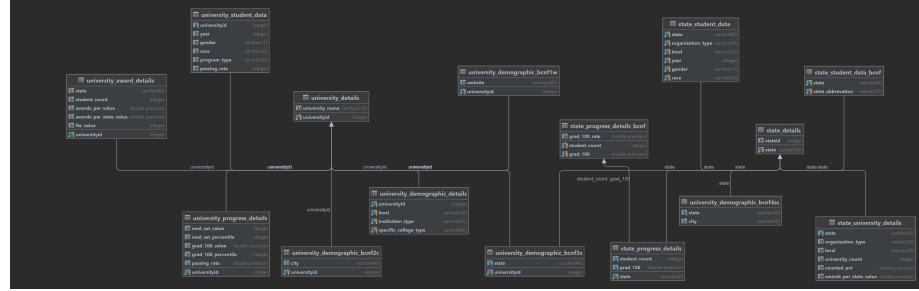
43
44
45 Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not
46 made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components
47 of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to
48 redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

49 © 2022 Association for Computing Machinery.

50 Manuscript submitted to ACM

51
52 Manuscript submitted to ACM

3 ER DIAGRAM

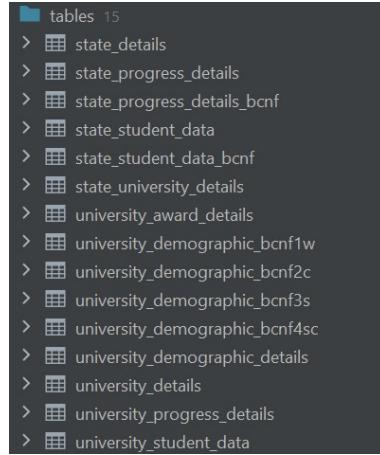


4 DATABASE IMPLEMENTATION

4.1 Data Schemas

In this section we will be presenting the schemas that are present in our dataset. In all this dataset includes nine tables or nine relational schemas.

The Following is the image of the tables from our pgadmin for our Dataset.



From the ER diagram we can see that the information about students that are currently enrolled in degree programs at Universities.

This table provides the student gender race distribution among the student by including attributes like gender and race. Most importantly, it includes the passing rate of the students, which serves as the primary quantifying metric for this dataset.

The above ER diagram includes information about the university's demographic data. This table provides the geographical data such as the state and the city in which a particular university is located. It also provides the type of institution which could be Public or Private. Under private it could be for profit or not for profit. The dataset also provides the website of such institutions for easy access to the details. The relation also has university id which servers as the unique identifier.

105 The above relation provides the information about the statewise information of the progress details for the institution
 106 information which has the type of program. Also it has grad_100 which shows the graduation as expected time. and
 107 grad_100_rate which is similar to the previous information but provides the rate as well.
 108

109 The relations shows how any state fairs as per the completion rate of student in their respective degree programs
 110 and levels and organization type. This relation provides a quantification of the state information.A detailed description
 111 of each attribute is give in the attributes section.
 112

113 This relation is used just to connect relations which either have stateid or state information.
 114

115 The relation like the previous relation which showed the state distribution of students and their completion rates,
 116 this provides the similar information with a hint of student demographic information.
 117

118 This relation is again a linking table with provides the university name with the appropriate university id which
 119 servers as a lookup table.
 120

121 The university awards table provides the statewise completion rate of students who have qualified their course in
 122 one attempt. The awards per value and awards per state value both provide the completion rate as per the state and
 123 the national average.The fte value provides the total full time graduates followed by a unique identifier that is the
 124 universityid.
 125

126 The University progress details provide very different information seen so far. This table provides the information
 127 about information about sat scores, sat percentiles the graduation rate followed by the graduation percentile rate with a
 128 unique identifier as the universityid.
 129

130
 131 **Relationships between tables** - The **university_details** the table contains university_name and universityid.
 132 The **university_awards_details** will contain all the information about awards for a particular university. **univer-**
 133 **sity_student_data** will contain demographic information about student data of that university. **university_progress_details**
 134 contains all the information about sat exam score eligibility, passing rate, etc for a particular university.**university_demographic_details**
 135 will contain details about the institution type, website, state, etc for a particular university. **state_details** contains stateid
 136 and statename for a particular state. **state_progress_details** contains data about program types offered, graduation
 137 rate, etc for a particular state. **state_university_details** will store different types of organizations, university count,
 138 and awards earned by universities in that state.**state_student_data** contains demographic details of student studying
 139 in that state.
 140

144 4.2 Primary and Foreign Keys

- 146 • The relation **university_details** has universityid is the primary key.
- 147 • The relation **state_details** has stateid is the primary key.
- 148 • The relation **University_student_data** has the universityid that serves as primary key. It is also the foreign
 149 key which is referred from relation **university_detail**.
- 150 • The relation **university_demographic_details** has the universityid that serves as the primary key. The foreign
 151 key of this relations is state and univesityid. state is refereed from **state_details** and univesityid is referred
 152 from **univesity_details**.
- 153 • The relation **state_student_data** has the state that serves as the foreign key which is referred from **state_details**.

- 157 • The relation **university_progress_detail** has the universityid that serves as primary key. It is also the foreign
158 key which is referred from relation **university_detail**.
- 159 • The relation **university_award_details** has the universityid that serves as primary key. It is also the foreign
160 key which is referred from relation **university_detail**.
- 161 • The relation **state_progress_details** has the state that serves as the foreign key which is referred from
162 **state_details**.
- 163 • The relation **state_university_details** has the state that serves as the foreign key which is referred from
164 **state_details**.
- 165
- 166

168 4.3 Data Source

170 The data for this project was collected from the following link below,

171 Data.world. 2022. College Completion - dataset by databeats. [online] Available at: <<https://data.world/databeats/college-completion>> [Accessed 16 October 2022] Manuscript submitted to ACM.

174 4.4 Deviation from Milestone-1

176 We have the following modification on the relations from checkpoint 1.

- 178 • **state_student_data** relation had the following attribute set{state, state_abbreviation, organization_type, level,
179 year, gender, race}. We have the following functional dependency in this relation which causes the relation to
180 not be in BC-NF form,

181 $\text{state} \rightarrow \text{state_abbreviation}$

182 $\text{state_abbreviation} \rightarrow \text{state}$

183 So, we decompose the relation into BCNF with the following candidate keys, state, organization_type, level, year, gender,
184 race state_abbreviation, organization_type, level, year, gender, race

185 Now we form two tables from state_student_data relation,

186 **state_student_data** and **state_student_data_bcnf**

- 188 • **state_progress_details** relation had the following attribute set {state, state_count, grad_100, grad_100_rate}. We have
189 the following functional dependency in this relation which causes the relation to not be in BC-NF form,

190 $\text{state} \rightarrow \text{state_count, grad100}$

191 $\text{state_count, grad100} \rightarrow \text{grad_100_rate}$

192 Now we also observe a transitive dependency, state can determine state_count grad 100, and state_count and grad 100
193 can determine grad_100_rate which causes transitive dependency.

194 Therefore, we decompose the relation into the following relations, **state_progress_details** and **state_progress_details_bcnf**

- 196 • **state_university_details** relation had the following attribute set {state, state_abbreviation, ,organization_type, level,
197 university_count, counted_pct, awards_per_state_value}. We have the following functional dependency in this relation
198 which causes the relation to not be in BC-NF form,

199 $\text{state} \rightarrow \text{state_abbreviation}$

200 $\text{state} \rightarrow \text{unicount}$

201 So, we decompose the relation into BCNF with the following candidate keys,

202 {state,organization_type, level, university_count, counted_pct, awards_per_state_value},

203 Manuscript submitted to ACM

209 {state_abbreviation,organization_type, level, university_count, counted_pct, awards_per_state_value}
 210 Now we form two tables from state_university_details relation,
 211 **state_university_details** and **state_university_details_bcnf**

- 212 • **university_demographic_details** relation had the following attribute set {city, state, level, institute_type, spe-
 213 cific_college_type, website, universityid} We have the following functional dependency in this relation which causes
 214 the relation to not be in BC-NF form,

215 universityid → website

216 universityid → city

217 universityid → state

218 So, we decompose the relation into the following BC-NF reduced tables.

219 **university_demographic_details**

220 **university_demographic_bcnf1w**

221 **university_demographic_bcnf2c**

222 **university_demographic_bcnf3s**

223 **university_demographic_bcnf4sc**

224 4.5 BCNF

- 225 • The relation **state_details** has no internal dependency and hence it is BCNF.

state_details	
	stateid integer
	state varchar(40)

226 The state is a candidate key that determines every other attribute.

227 The relations state_details do not have multi-valued attributes hence it is in 1NF.

228 The relations state_details do not have partial dependency hence it is in 2NF.

229 Additionally, we do not have a transitive dependency, and hence it is in 3NF.

230 Since we do not have mixed partial dependency, hence it is also in BC-NF.

- 231 • The relation **state_progress_details** has no internal dependency and hence it is BCNF.

state_progress_details	
	student_count integer
	grad_100 double precision
	state varchar(40)

261 State is a candidate key that determines every other attribute.
 262 The relations state_progress_details do not have multi-valued attributes hence it is in 1NF.
 263 The relations state_progress_details do not have partial dependency hence it is in 2NF.
 264 Additionally, we do not have a transitive dependency, and hence it is in 3NF.
 265 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 266
 267

- 268 • The relation **state_progress_details_bcnf** has no internal dependency, and hence it is BC-NF.
 269

state_progress_details_bcnf	
	grad_100_rate double precision
	student_count integer
	grad_100 double precision

277 grad_100 and student_count are candidate key that determines every other attribute.
 278 The relations state_progress_details_bcnf do not have multi-valued attributes hence it is in 1NF.
 279 The relations state_progress_details_bcnf do not have partial dependency hence it is in 2NF.
 280 Additionally, we do not have a transitive dependency, and hence it is in 3NF.
 281 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 282
 283

- 284 • The relation **state_student_data** has no internal dependency and hence it is BC-NF.
 285

state_student_data	
	state varchar(40)
	organization_type varchar(30)
	level varchar(20)
	year integer
	gender varchar(15)
	race varchar(20)

299 state, organization_type, level year, gender, race are candidate key.
 300 The relations state_student_data do not have multi-valued attributes hence it is in 1NF.
 301 The relations state_student_data do not have partial dependency hence it is in 2NF.
 302 Additionally, we do not have a transitive dependency, and hence it is in 3NF.
 303 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 304
 305

- 306 • The relation **state_student_data_bcnf** has no internal dependency and hence it is BCNF.
 307

state_student_data_bcnf	
	state varchar(40)
	state_abbreviation varchar(10)

state is the candidate key that determines every other state_abbreviation.

The relation state_student_data_bcnf do not have multi-valued attributes hence it is in 1NF.

The relation state_student_data_bcnf do not have partial dependency hence it is in 2NF.

Additionally, we do not have a transitive dependency and hence it is in 3NF.

Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_university_details** has no internal dependency and hence it is BCNF.

state_university_details	
	state varchar(40)
	organization_type varchar(30)
	level varchar(20)
	university_count integer
	counted_pct double precision
	awards_per_state_value double precision

state is a candidate key that determines every other attribute.

The relations **state_university_details** do not have multi-valued attributes hence it is in 1NF.

The relations state_university_details do not have partial dependency hence it is in 2NF.

Additionally, we do not have a transitive dependency, and hence it is in 3NF.

Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_award_details** has no internal dependency and hence it is BCNF.

university_award_details	
	state varchar(40)
	student_count integer
	awards_per_value double precision
	awards_per_state_value double precision
	fte_value integer
	universityid integer

universityid is the candidate key that determines every other attribute.

The relations university_award_details do not have multi-valued attributes hence it is in 1NF.

The relations university_award_details do not have partial dependency hence it is in 2NF.

Additionally, we do not have a transitive dependency and hence it is in 3NF.

365 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 366
 367

- 368 • The relation **university_demographic_bcnf1w** has no internal dependency and hence it is BCNF.

university_demographic_bcnf1w	
	website varchar(300)
	universityid integer

373 universityid is the candidate key that determines website.
 374

375 The relation **university_demographic_bcnf1w** do not have multivalued attributes hence it is in 1NF.
 376

377 The relations **university_demographic_bcnf1w** do not have partial dependency hence it is in 2NF.
 378

379 Additionally, we do not have a transitive dependency and hence it is in 3NF.
 380

381 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 382

- 383 • The relation **university_demographic_bcnf2c** has no internal dependency and hence it is BCNF.

university_demographic_bcnf2c	
	city varchar(40)
	universityid integer

387 universityid is the candidate key that determines city.
 388

389 The relations **university_demographic_bcnf2c** do not have multi-valued attributes hence it is in 1NF.
 390

391 The relations **university_demographic_bcnf2c** do not have partial dependency hence it is in 2NF.
 392

393 Additionally, we do not have a transitive dependency and hence it is in 3NF.
 394

395 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 396

- 397 • The relation **university_demographic_bcnf3s** has no internal dependency and hence it is BCNF.

university_demographic_bcnf3s	
	state varchar(40)
	universityid integer

401 universityid is the candidate key that determines state.
 402

403 The relations **university_demographic_bcnf3s** do not have multivalued attributes hence it is in 1NF.
 404

405 The relations **university_demographic_bcnf3s** do not have partial dependency hence it is in 2NF.
 406

407 Additionally, we do not have a transitive dependency and hence it is in 3NF.
 408

409 Since we do not have mixed partial dependency, hence it is also in BC-NF.
 410

- 411 • The relation **university_demographic_bcnf4sc** has no internal dependency and hence it is BCNF.

university_demographic_bcnf4sc	
	state varchar(40)
	city varchar(40)

417 State here is the candidate key which can determine the city of university.

418 The relations university_demographic_bcnf4sc do not have multivalued attributes hence it is in 1NF.

419 The relations university_demographic_bcnf4sc do not have partial dependency hence it is in 2NF.

420 Additionally, we do not have a transitive dependency and hence it is in 3NF.

421 Since we do not have mixed partial dependency, hence it is also in BC-NF.

- 422
- 423 • The relation **university_demographic_details** has no internal dependency and hence it is BCNF.

university_demographic_details	
	universityid integer
	level varchar(20)
	institution_type varchar(50)
	specific_college_type varchar(80)

424 Universityid,level,institution_type,specific_college_type together here is the candidate key which can deter-
425 mined.

426 The relations university_demographic_details do not have multivalued attributes hence it is in 1NF.

427 The relations university_demographic_details do not have partial dependency hence it is in 2NF.

428 Additionally, we do not have a transitive dependency and hence it is in 3NF.

429 Since we do not have mixed partial dependency, hence it is also in BC-NF.

- 430
- 431 • The relation **university_details** has no internal dependency and hence it is BCNF.//

university_details	
	university_name varchar(200)
	universityid integer

432 Universityid here is the candidate key which can determine the university_name.

433 The relations university_details do not have multivalued attributes hence it is in 1NF.

434 The relations university_details do not have partial dependency hence it is in 2NF.

435 Additionally, we do not have a transitive dependency and hence it is in 3NF.

436 Since we do not have mixed partial dependency, hence it is also in BC-NF.

- 437
- 438 • The relation **university_progress_details** has no internal dependency and hence it is BCNF.

469		
470		
471		
472		
473		
474		
475		
476		
477		
478		
479		
480		
481		university_progress_details
482		med_sat_value integer
483		med_sat_percentile integer
484		grad_100_value double precision
485		grad_100_percentile integer
486		passing_rate double precision
487		universityid integer
488		
489		

Universityid here is the candidate key which can determine the other attributes of table.

The relations university_progress_details do not have multivalued attributes hence it is in 1NF.

The relations university_progress_details do not have partial dependency hence it is in 2NF.

Additionally, we do not have a transitive dependency and hence it is in 3NF.

Since we do not have mixed partial dependency, hence it is also in BC-NF.

4.6 Attributes Information

- From the ER diagram we can see that from the relation **university_student_data**. The attributes included in this schema are the **universityid** that serves as the unique identifier for students in this database. The universityid has datatype as *integer*. The attribute **year** shows the enrollment year for a particular student which is of an *integer* datatype. The attributes **gender** and **race** are both *varchar* with sizes 15 and 30 respectively. The attribute **student_count** involves the number of students which can be one of the following bachelor's degrees, associate degrees, and certificate programs and has a datatype *varchar* with size 20. The attribute **passing_rate** shows the primary metric of this dataset which is the particular students passing rate which has the datatype as *integer*.
- The relation **university_demographic_details** provides the following attributes. The attribute **city** and **state** provides the location details has the datatype *varchar* 40. the attribute **level** has the datatype *varchar* 20. **institution_type** and **specific_college_type** are both *varchar* with sizes 50 and 80. **website** gives the university website which has the datatype as *varchar* with size 300. Finally **universityid** uniquely identifies the the type of university which has the datatype *integer*.
- state_progress_details** has the attributes **state** which has the datatype as *varchar* of size 40 followed by the **program_type** which is also of datatype *varchar* with size 30. The attributes **grad_100** and **grad_100_rate** both are *float8* datatype which provides the information about how the student pass their program ideally and their rates.
- The relation **state_university_details** has the attributes **state** which has the datatype *varchar* with size 40 followed by its shorthand **state_abbreviation** of type *varchar* of size 10. **organization_type** which specifies if the university is public or private. If it is private either it is for profit or not for profit. The data type for this is *varchar* with size 30. Now the attribute **level** specifies if it's a 4-year degree program or a 2-year degree program which has datatype *varchar* of size 20. **university_count** specifies how many of such universities are present in a particular stateid which has the datatype *integer*. The attribute **counted_pct** has the information

521 about how many first time student have entered the degree program which has the datatype *float8*. The
 522 **awards_per_state_value** has the datatype *float8* shows how many of the students who started the degree
 523 have complete the degree at their first attempt.

- 524 • **state_details** is only used to connect tables which have only stateid and tables which have states.
- 525 • The relation **state_student_data** has the attributes state which has the datatype *varchar* which serves as an
 526 identifier of this relation. Furthermore, we have the shorthand **state_abbreviation** for a state which was also
 527 there in a previous relation with datatype *varchar*. The **level** provides the level of education which can be a
 528 4-year or 2-year program. Additionally, we have the **organization_type** which specifies if the if it is a public,
 529 private for profit, or not as mentioned in earlier relations as well. The attributes **gender** and **race** of type
 530 *varchar* provide the information state-wide followed by the year of induction for the cohort/student.
- 531 • The next relation is the **university_details** servers as a lookup for university names and universities its similar
 532 to the state details.
- 533 • The relation **university_award_details** is the distribution of the completion rate with the attributes **awards_per_value**
 534 of type *float8* **awards_per_state_value** of type *float8* and the **fte_value** of type *integer* which serves as the
 535 indicators for completion success-rate with respect to state, national and total completion rates. Also, the
 536 **universityid** is the unique identifier with type *integer* as mentioned earlier.
- 537 • **university_progress_details** is very different from the previous attributes with has a quantifier to the comple-
 538 tion rates based on the SAT scores **med_sat_value** which is of type *integer* followed by **med_sat_percentile**
 539 which is of type *integer* which provides percentile quantification, which is followed by the **grad_100_value**
 540 and **grad_100_percentile** of types *float8* and **passing_rate** of type *integer*, all three of them serve the purpose
 541 of defining the progress of a university on the basis of the completion rate of student undertaking the degree
 542 program with the **universityid** which links this to the respective universities. All other tables are BC-NF forms
 543 created by decomposing the core eight tables defined in checkpoint-1 submission.
- 544
- 545
- 546
- 547
- 548
- 549

5 USER INTERFACE-UI

557 The following is our welcome page,

558 *Welcome to College Completion Database(SMG)*

559 Welcome to the State Insertion

560 What do you want to do?

561 SHOW STATE DATA

562 INSERT DATA

563 DELETE DATA

564 UPDATE DATA

568 We initially have the following data in our user interface,

573	Nevada	32
574	New Hampshire	33
575	New Jersey	34
576	New Mexico	35
577	New York	36
578	North Carolina	37
579	North Dakota	38
580	Ohio	39
581	Oklahoma	40
582	Oregon	41
583	Pennsylvania	42
584	Rhode Island	44
585	South Carolina	45
586	South Dakota	46
587	Tennessee	47
588	Texas	48
589	Utah	49
590	Vermont	50
591	Virginia	51
592	Washington	53
593	West Virginia	54
594	Wisconsin	55
595	Wyoming	56
596	Gujarat	88
597	Punjab	188

608 [Back To HOME!!!](#)

609 Now we show the CRUD operations,
 610 We create an entry for **Kerala** with state-ID as **500**.
 611 Here is the UI operation,

```

625
626 State Name: 
627
628 State ID: 
629 
630
631
632 
633
634 
635
636
637

```

The result of the data after the operation

New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Punjab	188
Kerela	500

[Back To HOME!!!](#)

We Update the record **Kerala** to have state-ID as **999**.

Here is the UI operation,

```

677
678 Insert State Details to UPDATE
679
680 State Name: 
681 State ID:  
682
683
684
685 
686
687
688
689 
690
691
692 The result of the data after the operation
693
694 

|                |     |
|----------------|-----|
| Nevada         | 32  |
| New Hampshire  | 33  |
| New Jersey     | 34  |
| New Mexico     | 35  |
| New York       | 36  |
| North Carolina | 37  |
| North Dakota   | 38  |
| Ohio           | 39  |
| Oklahoma       | 40  |
| Oregon         | 41  |
| Pennsylvania   | 42  |
| Rhode Island   | 44  |
| South Carolina | 45  |
| South Dakota   | 46  |
| Tennessee      | 47  |
| Texas          | 48  |
| Utah           | 49  |
| Vermont        | 50  |
| Virginia       | 51  |
| Washington     | 53  |
| West Virginia  | 54  |
| Wisconsin      | 55  |
| Wyoming        | 56  |
| Gujarat        | 88  |
| Punjab         | 188 |
| Kerela         | 999 |


```

729
 730 **Insert State Details to Delete**
 731

732 State Name:

733 State ID:

734

736

739

741 We Delete an entry for **Punjab** with state-ID as **188**.
 742

743 The result of the data after the operation.

Nebraska	31
Nevada	32
New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Kerela	999

774 [Back To HOME!!!](#)
 775

776
 777 **6 QUERY EXECUTION**
 778

779 In this section, we include four queries. Additionally, we also include two trigger queries
 780

- The Problem statement for the following query is given below, We find the University id and University name with the Highest number of students. This query is useful as it provides insight into choosing the most popular universities among students.

```

781      --Query 1--> To find the University id and University name with highest number of students.
782      EXPLAIN ANALYSE select university_details.universityid,
783                           university_details.university_name,
784                           university_award_details.student_count
785                           from university_details
786                           join university_award_details
787                           ON university_award_details.universityid = university_details.universityid
788                           where university_award_details.student_count<
789                               (select max(ui.student_count)
790                                from university_award_details ui);
791

```

- The Problem statement for the following query is given below, We find the total number of Males in California along with their stateid. This is a sample query to check the gender diversity at universities in California.

```

792      -- Query 2--> Find total number of Male in California along with their stateid.
793      EXPLAIN ANALYSE SELECT state_details.stateid,state_details.state,
794                           state_student_data.gender,COUNT(state_student_data.gender)
795                           AS Gender_count
796                           FROM state_details JOIN state_student_data USING(state)
797                           WHERE state_details.data.gender='Male'
798                           AND state_details.state='California'
799                           GROUP BY state_details.stateid,state_details.state,state_student_data.gender;
800

```

- The Problem statement for the following query is given below, We find the specific college type starting with research universities, and find the universityid,name of university along with passing rate in decreasing order.

```

801      -- --Query 3--> For specific college type starting with research universities,
802      -- -- find the universityid,name of university along with passing
803      -- -- rate in decreasing order.
804      -- -->
805      -->
806      --> EXPLAIN ANALYSE SELECT university_details.universityid,university_details.university_name,
807                           university_details.university_type,
808                           university_progress_details.passing_rate
809                           FROM university_details,university_demographic_details,
810                           university_progress_detail
811                           WHERE university_details.universityid = university_demographic_details.universityid
812                           AND university_details.university_type = university_progress_details.universityid
813                           AND university_demographic_details_specific.college_type LIKE 'Research'
814                           ORDER BY university_progress_details.passing_rate DESC;
815

```

- The Problem statement for the following query is given below, Find states having more than 50 universities in descending order based on universities count

```

816      -- -- query 4--> Find states having more than 50 universities in their descending order based on
817      -- -- universities count.
818      -- -->
819      -->
820      --> EXPLAIN ANALYSE SELECT university_award_details.state,
821                           count(university_details.universityid)
822                           FROM university_award_details
823                           INNER JOIN university_details
824                           ON university_details.universityid = university_award_details.universityid
825                           GROUP BY university_award_details.state
826                           HAVING count(university_details.universityid) >50
827                           ORDER BY count(university_details.universityid) DESC ;
828

```

- The Problem statement for the following query is given below, Creating a trigger 1.

```

829      -- Trigger 1 for insertion of records in student_university_details & in state_student_data
830      -- for each row execute procedure Insert_records();
831      CREATE TRIGGER state_univ_Insert AFTER INSERT ON state_student_data
832      FOR EACH ROW EXECUTE PROCEDURE Insert_records();
833
834      CREATE OR REPLACE FUNCTION Insert_records() RETURNS TRIGGER AS $state_univ_Insert$
835      BEGIN
836          INSERT INTO state_university_details VALUES (NEW.state,NEW.organization_type,NEW.level);
837          RETURN NEW;
838      END;
839      $state_univ_Insert$ LANGUAGE plpgsql;
840
841      SELECT * FROM state_university_details;
842      SELECT * FROM state_student_data;
843
844      INSERT INTO state_student_data VALUES('Alaska','Presidency','2-year','20','32.1','78.0');
845

```

Now, we insert data to the state_student_data table where the trigger is created.

```

846      insert into state_student_data values('New York','University at Buffalo','2-year','20','32.1','78.0')
847      Data output Messages Notifications
848      INSERT n 1
849      Query returned successfully in 73 msec.
850

```

Insertion successful on table state_student_data where the Trigger is created.

```

851      insert into state_student_data values('New York','University at Buffalo','2-year','20','32.1','78.0'
852      SELECT * FROM state_university_details WHERE organization_type='University at Buffalo';
853      SELECT * FROM state_student_data WHERE organization_type='University at Buffalo';
854

```

	state	organization_type	level	year	gender	race
1	New York	University at Buffalo	2-year	20	32.1	78.0

Insertion is done to the reference child table in Trigger Function. Data is displayed as above with three columns referenced from a parent but the other three columns left as NULL.

```
833 SELECT * from state_university_details where organization_type='University'@
834 SELECT * from state_student_data where organization_type='University'@ Buffalo';
835 Data output Messages Notifications
836
837
838
839 • Creating a trigger 2.
840
841 TRIGGER $univid_insert BEFORE INSERT ON state_student_data
842 CREATE TRIGGER $univid_insert AFTER Insert ON university_details
843 FOR EACH ROW EXECUTE PROCEDURE Insertrecords();
844
845
846 CREATE OR REPLACE FUNCTION Insertrecords() RETURNS TRIGGER AS $univid_insert$
847 BEGIN
848   INSERT INTO university_demographic_bcnfval VALUES (NEW.university_id);
849   RETURN NEW;
850 END;
851
852 $univid_insert$ LANGUAGE plpgsql;
853
854
855 SELECT * from university_details order by university_id;
856 SELECT * from university_demographic_bcnfval order by university_id;
```

insertion into University_details, where the trigger is created.

```
847     INSERT into university_details values(10234,'University at Buffalo');
848     delete from university_details where universityId = 123;
849
850     Data output  Messages  Notifications
851
852     INSERT 0 1
853
854     Query returned successfully in 66 msec.
```

Displaying the University_details table to verify that insertion is done for this table where we have created the triggers.

```
854          SELECT * from university_details order by universityId;
855          SELECT * from university_descriptive_bchfw order by universityId;
856
857          Data output  Messages  Notifications
858
859          +-----+-----+
860          |universityId|university_name|
861          +-----+-----+
862          |1          |University of Buffalo |
863          |2          |Alabama A&M University |
864          |3          |University of Alabama ... |
865          |4          |Amarillo University |
866          |5          |University of Alabama ... |
867          |6          |Alabama State Univers... |
868
```

As we have referenced to university_demographic_bcnf1w in the trigger created Function, so now displayed the data to verify whether it's successfully inserted in the child table.

```
863 SELECT * from university_demographic_bcnfw order by universityId;
864 Data output | Messages | Notifications
865 

|   | universityId | website                 |
|---|--------------|-------------------------|
| 1 | 10234        | www.smu.edu             |
| 2 | 10554        | www.sas.upenn.edu       |
| 3 | 10640        | www.psu.edu             |
| 4 | 106990       | www.arizonauniversity.. |
| 5 | 107006       | www.uw.edu              |
| 6 | 107274       | www.alasu.edu/emaU..    |


870 END;
```

Similarly, We have trigger the Updation query and Trigger deletion Query.

- Trigger Update

```
871
872    -- Update trigger query --
873    CREATE TRIGGER update_state AFTER UPDATE ON state_details
874        for each row execute procedure Update_records();
875
876    CREATE OR REPLACE FUNCTION Update_records() RETURNS TRIGGER as $update_state$
877        BEGIN
878            insert into university_demographic_bcnf4sc VALUES (new.state);
879            RETURN NEW;
880        END;
881        $update_state$ LANGUAGE plpgsql;
882
883    -- Update Query --
884    update state_details set state='Karnataka' where stateid=100;
885
886    -- Showing tables --
887    select * from state_details order by stateid DESC;
888    select * from university_demographic_bcnf4sc where state like 'Kar%';
```

- Trigger Deletion

Manuscript submitted to ACM

```

885   -- delete Trigger for state
886   CREATE TRIGGER delete_state AFTER DELETE ON state_details
887     for each row execute procedure Delete_records();
888
889   CREATE or REPLACE FUNCTION Delete_records() RETURNS TRIGGER as $delete_state$
890   BEGIN
891     DELETE from university_demographic_bcnf4sc where state=new.state;
892     RETURN NEW;
893   END;
894   $delete_state$ LANGUAGE plpgsql;
895
896   -- delete query
897   delete from state_details where state like 'Karn%';
898
899
900
911
922

```

7 QUERY OPTIMIZATION

For the queries mentioned in the previous section, we introduce indexing to optimize its performance.

For the first query, we create an index for university details and university_award_details.

```

-- Query1==>
EXPLAIN ANALYSE select university_details.universityid,
university_details.university_name,university_award_details.student_count
from university_details
join university_award_details
ON university_award_details.universityid = university_details.universityid
where university_award_details.student_count=-
      select max(u1.student_count)
      from university_award_details u1);
-- Query optimisation to improve performance on query 1
CREATE INDEX universityDetails on university_details(universityid,university_name);
CREATE INDEX universityawardDetails on university_award_details(universityid,state);

```

We can see the performance increase with the index implementation.

The figure consists of two vertically stacked screenshots of the pgAdmin interface. The top screenshot shows the original query and its execution plan, which is very inefficient with a cost of 82.77 and many loops. The bottom screenshot shows the same query after creating two indexes (universityDetails and universityawardDetails). The execution plan is now highly optimized, with a cost of 0.278 ms and a single nested loop, indicating significant performance improvement.

Fig. 1. Query Result 1

For the second query, we create an index for StateDetails and states_student_data.

```

-- Query 2 ==> Find total number of Male in California along with their stateid.
EXPLAIN ANALYSE SELECT state_details.stateid,state_details.state,
state_student_data.gender,COUNT(state_student_data.gender)
AS Gender_count
FROM state_details JOIN state_student_data USING(state)
WHERE state_student_data.gender='Male'
AND state_details.state='California'
GROUP BY state_details.stateid,state_details.state,state_student_data.gender;
-- Query optimisation to improve performance on query 2
CREATE INDEX StateDetails on state_details(stateid,state);
CREATE INDEX statestudentdata on state_student_data(gender);

```

We can see the performance increase with the index implementation.

The figure consists of two vertically stacked screenshots of the pgAdmin interface. The top screenshot shows the original query and its execution plan, which is very inefficient with a cost of 82.77 and many loops. The bottom screenshot shows the same query after creating two indexes (StateDetails and statestudentdata). The execution plan is now highly optimized, with a cost of 0.241 ms and a single group aggregate loop, indicating significant performance improvement.

Fig. 2. Query Result 2

For the third query, we create an index for universityDemographicDetails and universityProgressDetails

```

937
938
939 -- Query 3 => For specific college type starting with research universities,
940   Find the universityid, name of university along with passing
941   rate, demographic details
942
943 EXPLAIN ANALYSE SELECT university_details.universityid,university_details.university_name,
944   university_demographic_details.specified_college_type,
945   university_progress_details.passing_rate
946   FROM university_details,university_demographic_details,
947   university_progress_details
948   WHERE university_details.universityid = university_demographic_details.universityid
949   AND university_demographic_details.specified_college_type LIKE 'Research'
950   ORDER BY university_progress_details.passing_rate DESC;
951
952 -- Query optimization to improve performance on query 3
953 CREATE INDEX universityDemographicDetails ON university_demographic_details(specified_college_type);
954 CREATE INDEX universityProgressDetails ON university_progress_details(passing_rate);
955
956 drop index universitydemographicdetails;
957 drop index universityprogressdetails;

```

We can see the performance increase with the index implementation.

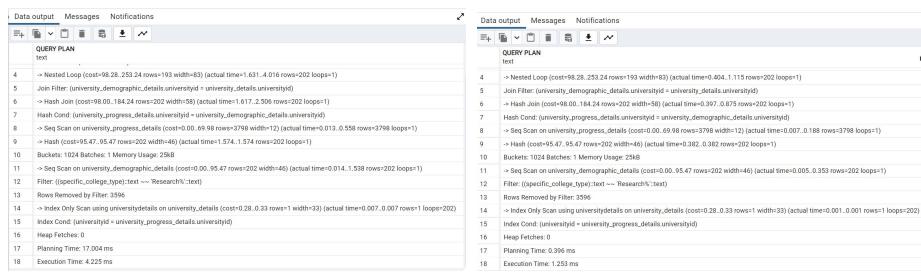


Fig. 3. Query Result 3

For the fourth query, we create an index for university details and university_award_details.

```

961
962
963 -- query 4 => find states having more than 50 universities in their descending order based on
964   number of universities
965
966 EXPLAIN ANALYSE SELECT university_award_details.state,
967   count(university_details.universityid)
968   AS Universities_Count
969   FROM university_award_details
970   INNER JOIN university_details
971   ON university_details.universityid = university_award_details.universityid
972   GROUP BY university_award_details.state
973   HAVING count(university_details.universityid) >= 50
974   ORDER BY count(university_details.universityid) DESC ;
975
976 CREATE INDEX universityDetails ON university_details(universityid,university_name);
977 CREATE INDEX universityAwardDetails ON university_award_details(universityid,state);

```

We can see the performance increase with the index implementation.

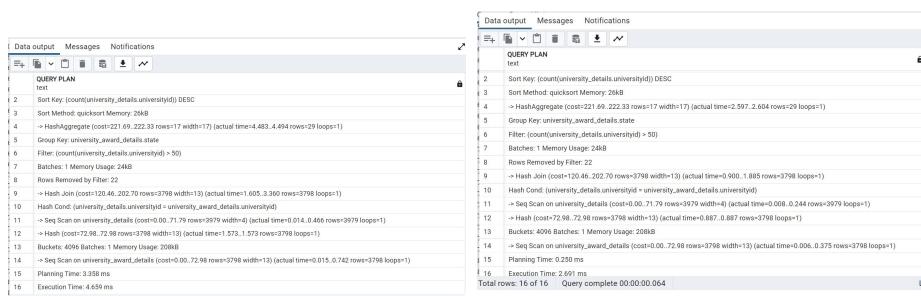


Fig. 4. Query Result 4

8 FUTURE SCOPE

The database for the student completion data-set is still in its nascent phases. This effort of making student degree completion rate public started by Bill and Melinda Gates foundations could be carried on by universities and governments

989 We can finally summarize the Query Optimization from the table below.
 990
 991

Table 1. Query performance

Query #	Performance Before in ms	Performance After in ms	Percentage Decrease in time
1	0.830	0.633	23.7349%
2	0.390	0.241	38.2051%
3	4.225	1.253	70.3432%
4	4.659	2.691	42.2408%

992 alike. Makings such data-sets publicly available makes not-only student make better choices but also promotes better
 993 recourse allocation by universities and state government. Additionally, this data-set being maintained by governing
 994 body, could promote better problems beings discussed and can give rise to understanding college dropout rates and
 995 address such issues in a more practical way.

1004 9 REFERENCES

- 1006 • Data.world. 2022. College Completion - dataset by databeats. [online] Available at: <<https://data.world/databeats/college-completion>> [Accessed 16 October 2022]